Volume 1, Number 2, 2012

International Journal of Research in Industrial Engineering

Journal homepage: www.nvlscience.com/index.php/ijrie

Duplicate Genetic Algorithm for Scheduling a Bi-Objective Flexible Job Shop Problem

R. Tavakkoli-Moghaddam[1], N. ShahsavariPour[2], H. Mohammadi-Andargoli[3,*], M.H. Abolhasani-Ashkezari[3]

[1] Department of Industrial Engineering, College of Engineering, University of Tehran,Tehran, Iran
[2] Department of Industrial Management, Vali-e-Asr University, Rafsanjan, Iran
[3]Department of Industrial Engineering, Science and Research Branch, Islamic Azad University, Kerman, Iran

ARTICLE INFO

Article history :
Received:
May 20, 2012
Revised:
August 10, 2012
Accepted:
September 25, 2012

Keywords:
Flexible job shop scheduling problem, Duplicate genetic algorithm, Bi-objective optimization.

ABSTRACT

This paper addresses the permutation of a flexible job shop problem that minimizes the makespan and total idleness as a bi-objective problem. This optimization problem is an NP-hard one because a large solution space allocated to it. We use a duplicate genetic algorithm (DGA) to solve the problem, which is developed a genetic algorithm procedure. Since the proposed DGA is working based on the GA, it often offers a better solution than the standard GA because it includes the rational and appropriate justification. The proposed DGA is used the useful features and concepts of elitism and local search, simultaneously. It provides local search for the best solution in every generation with the neighborhood structure in several stages and stores them in an external list for reuse as a secondary population of the GA.The performance of the proposed GA is evaluated by a number of numerical experiments. By comparing the results of the DGA other algorithms, we realize that our proposed DGA is efficient and appropriate for solving the given problem.

1. Introduction

The importance of scheduling has increased in the recent decades because of increasing customer demands for a variety and changing market in according global competition. In addition, it has great impact in all domains of manufacturing, management and utilization. Today, most scheduling problems are complex optimization problems that are too difficult to solve. The job shop scheduling problem (JSSP) is one of the most popular manufacturing optimization models used in practice [1]. It is well known that the JSSP is NP-hard [2]. Hence, general and deterministic methods of search are inefficient in problems with a larger size. The classical JSSP consists of scheduling a set of jobs on a set of machines with the objective to minimize a certain criterion, subject to each job is to be processed on each machine in a pre-defined sequence. Each operation of a job is to be processed only on one machine at a time, so different heuristic and meta-heuristic algorithms are considered for solving JSSPs.

*Corresponding author
E-mail address: hamed.mohammadi327@gmail.com

The flexible jobshop scheduling problem (FJSSP) is an extension of the jobshop scheduling problem (JSSP) that allows an operation to be processed by any machine from a given set along different routes. In other hand, the FJSSP is a generalization of the job shop and the parallel machine environment providing a closer approximation to a wide range of real manufacturing systems, in which there are a set of work centers in a flexible job shop. Each work center has a set of parallel machines with possibly different efficiency. An operation can be performed by any machine in a work center. Consequently, this results two sub-problems: the first one is the routing problem (i.e., the assignment of operations to machines) that assigns each operation to a machine among a set of machines authorized for each job, and the second one is the scheduling problem (i.e., determining the starting time of each operation) that involves sequencing the operations assigned to the machines in order to obtain a feasible schedule minimizing a or several predefined objective. The FJSSP is NP-hard since it is an extension of the JSSP that has been proven to be NP-hard [2]. Furthermore, instead of considering only a single objective, most scheduling problems in practice involve optimization of several competing objectives, simultaneously. Therefore, in order to cope with the FJSSPs found in practice, efficient optimization techniques are required to deal with both bi-objectives and exponential search space complexity. The literature of FJSSPs is considerably sparser than the literature of JSSPs. The research on the bi-objective FJSSP is much less than the mono-objective FJSSP.

Bruker and Schlie [3] developed a polynomial algorithm for solving the FJSSP with two jobs. For solving the realistic case with more than two jobs, two types of approaches have been used, namely hierarchical and integrated approaches. In hierarchical approaches, the assignment of operations to machines and the sequencing of operations on the resources or machines are treated separately, whereas in integrated approaches, the assignment and sequencing are not differentiated. Brandimarte [4] was the first to use decomposition for the FJSP. He solved the routing sub-problem using some existing dispatching rules and then focused on the scheduling sub-problem using tabu search (TS); whereas in integrated approaches, the assignment and sequencing are not differentiated. Hurink et al. [5] proposed (TS). They considered the reassignment and rescheduling as two different types of moves. Chambers [6] developed TS to solve the problem. Dauzere-Peres and Paulli [7] proposed an integrated approach to define a neighborhood structure for the problem, in which there is no distinction between re-assigning and re-sequencing an operation. Tung et al. [8] developed a similar approach for scheduling a flexible manufacturing system. Mastrolilli and Gambardella [9] proposed some neighborhood functions for the FJSSP, which could be used in meta-heuristic optimization techniques. Mati et al. [10] proposed a greedy heuristic to deal simultaneously with assigning and sequencing sub-problems of the FJSSP. The advantage of their heuristic is its ability to take into account the assumption of identical machines. Parsopoulos and Vrahatis [11] conducted the first research on particle swarm optimization (PSO) in multi-objective optimization problems.

Kacem et al. [12-13] used a GA for two multi-objective approaches using either of the weighted summation of objectives or Pareto approaches. The first one (Kacem et al.) is controlled by the assigned model generated through approach by localization and the second

one (Kacem et al.) used a hybridization of evolutionary and fuzzy logic algorithms to solve the FJSSP. Rigao [14] developed two heuristics based on TS, namely a hierarchical procedure and a multiple start procedure. Wu and Weng [15] considered the problem with job earliness and tardiness objectives, and proposed a multi-agent scheduling method. Xia and Wu [16] treated this problem with a hybrid of PSO and simulated annealing (SA) as a local search algorithm. Zhang and Gen [17] proposed a multi-stage operation-based GA to deal with the FJSSP from a point view of dynamic programming. Gao et al. [18] developed a new approach hybridizing GA with variable neighborhood descent to exploit the "global search ability" of the GA and "the local search ability" of variable neighborhood descent for solving multi-objective problems. Zhang et al. [19] proposed hybridizing the two meta-heuristics, namely PSO and TS, to solve the multi-objective FJSSP. Tay and Ho [20] investigated the potential use of genetic programming (GP) for evolving effective and robust composite dispatching rules for solving the multi- objective FJSSP to minimizes the makespan, mean tardiness, and mean flow time. Xing et al. [21] presented a simulation model framework to solve the multi-objective FJSSP by a weighted summation method. They improved the sequencing performance by using ant colony optimization algorithm (ACO). In addition, Xing et al. [22] developed their previous approach and proposed an efficient search method for the problem.

In fact, the purpose of this paper is to optimize the JSSP with bi-objectives. The objectives are to minimize the makespan and total idleness on the basis of our new approach, namely duplicate genetic algorithm (DGA). This algorithm provides a local search for the best solution in every generation with the determined neighborhood structure in several stages and stores them in an external list for reuse as secondary population genetic algorithm. In contrast to other methods, the second objective covers other objectives described in the related literature. Thus, it leads to an optimal solution. The remainder of this paper is organized as follows. Section 2 introduces a specific formal problem definition. In Section 3, we illustrate the strategy of an optimization approach of our proposed DGA. In Section 4, we use some numerical examples to show the advantage of this algorithm. Finally, the conclusion is discussed in Section 5.

2. Problem definition

The FJSSP has been described in many papers and literatures and its general framework is as follows. There are n jobs that should be processed on a group of m machines in a way that each i job $(1 \le i \le n)$involves a sequence of n_i operation such as $O_{i,1}, O_{i,2}, ..., O_{i,n}$ where $O_{i,k}$ (K-th operation of job i) should be processed without interruption on a predefined machine $M_{i,k}$ during $P_{i,k}$ time units out of a set of given compatible machines, such as $M = \{M_1, M_2, ..., M_m\}$. The operations $O_{i,1}, O_{i,2}, ..., O_{i,n}$ should be processed one after another in the given order and each machine can process at most one operation at a time. Referring to Shao et al. [19] and Gao et al. [23], the FJSSP is needed to determine both an assignment and a sequence of the operations on the machines in order to optimization the

objectives. Therefore, a bi-objective FJSSP is formulated as the following overview by Tay and Ho [20]:

- Let $J = \{J_i\}_{1 \le j \le n_i}$ indexed i be a set of n_i jobs to be scheduled. Each job J_i consists of a predetermined sequence of operations. Let be $O_{i,k}$ operation the *K*-th of J_i.
- Let $M = \{M_z\}_{1 \le z \le m}$, indexed z, be a set of m machines.
- Each machine can process only one operation at a time.
- Each operation $O_{i,k}$ can be processed without interrupting on one of a set of machines M_z in a given set with $p_{i,k,z}$ time units.

During the process of solving this problem, the following assumptions are made:

- Each operation cannot be interrupted during its performance (non-preemptive condition);
- Setting up times of machines and move times between operations are negligible,
- Machines are independent from each other and jobs are also independent from each other.
- At a given time, a machine can only execute one operation. It becomes available to other operations only if the operation, which is processing, is completed.
- There are no precedence constraints among the operations of different jobs.

A general multi-objective minimization problem can be defined as: minimizing a function $f(x)$, with $p, (p \succ 1)$ decision variables and Q objectives $(Q \succ 1)$, subject to several equality or inequality constraints. More precise definitions of the terms about the multi-objective optimization can be found in Deb [24].

$$\underset{x \in \Omega}{\text{Min}} \quad f(x) = \left(f_1(x), f_2(x), ..., f_q(x), ..., f_Q(x)\right) \tag{1}$$

whereas Ω is the feasible solution space and $x = \{x_1, x_2, ..., x_p, ...x_P\}$ the set of p-dimensional decision variables (continuous, discrete or integer), i.e., a possible solution for the considered problem; $f_q(x)$ is the q-th objective function $(1 \prec q \prec Q)$.In this study, we minimize two objectives as follows:

$f_1(x)$: Makespan of the jobs (C_M) or the maximal completion time of machines.

$f_2(x)$: Total idleness of the machines, which represents the total idleness time over all machines.

The makespan is the maximum completion time of machines (i.e., the time difference between the first workstation to the last workstation means). Also, the summation of idleness time on all machines to finish the last operation from end job means total idleness time. The two objectives in the definition are the typical objectives in production scheduling that

frequently trade-off against each other. For instance, a schedule that satisfies the minimization of makespan can lead to a larger total idleness. Therefore, the ideal schedule is a schedule that produces trade-offs among different objectives. We focus our presentation on evolutionary approaches that can be classified into three types [25]:

- Transmission of multi-objective problem into a mono-objective by a weighted sum of the objectives combining.
- The processing of different objectives in a separated way.
- The Pareto approach is directly based on the Pareto optimization concept. It aims at satisfying two goals, namely converging to the Pareto front and obtaining the diversified solutions scattered all over the Pareto front.

The objective function in this paper is based on the first type. The weighted sum of the mentioned bi-objective values $\left(f_{1(x)}, f_{2(x)}\right)$ is taken as the objective function $F(l)$.

3. Proposed duplicate genetic algorithm

In this paper, the proposed approach is to use the duplicate genetic algorithm (DGA). It will include a stronger search from other algorithms (e.g., GA, SA, and TS), because of using local search and a secondary search with property of elitism. First, the initial population is generated randomly in the GA. Then, the best solution is identified in each generation of the GA as a semi-active solution. The semi-active solution evolves by using TS in a certain neighborhood structure as an active solution. The secondary population of the DGA is composed of all active solutions. Finally, we re-use the GA for the convergence the secondary population. Therefore, this process is called DGA, which property of the elitism will be much as possible and the search will achieve in the more space of solutions. In following, we present the framework of our proposed DGA.

Remark 1: In the final generations, which the GA closes to the optimum or approximately optimum solution, the variety of a semi-active solution is reduced in the algorithm because of convergence of the genetic algorithm. This means that a semi-active solution in the last generations of the GA has a greater weight than the first generations in the proposed DGA. This weight is controlled by the GA, even without the direct intervention of a parameter.

3.1. Genetic algorithm

Holland [26] first introduced the standard GA. It is research designed to solve NP-hard problems in a reasonable time. The basic idea of these algorithms is to simulate the selection of the scientific natural phenomena. The GA is a population-based evolutionary algorithm inspired by the behavior of the cells. In the GA, the population evolves based on the principle of natural selection (i.e., the survival of most qualified individuals) in the several generations. Thus, the GA finding the best solution goes ahead in the set of solutions.

3.1.1. Solutions encoding and chromosome structure

Showing the structure of a solution, called as chromosome, is important. An appropriate chromosome leads to the correct guidance of problem solving. Therefore, A solution of the FJSSP needs two sequences, namely assignment of operations to machines and sequence of operations on the machines. The scheme of the used chromosome can be represented by the structure shown in Fig.1. The first vector (ν_1) provides an assignment of operations to machines and other vector (ν_2) provides a sequence of operations on the machines. This figure shows an instance of a structure of the chromosome that it has two dimensions. It is drawn for the FJSSP with three jobs and three machines and nine operations $3\times 3/9$.

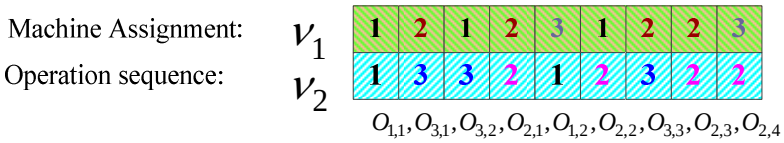

Fig. 1. Structure of a solution for the FJSSP with $3\times 3/9$

We generate the initial population in the GA randomly in size p .Of course, for accelerating the problem of solving time, we apply the GA framework in a string mode for each chromosome. Therefore, we have a sequencing related to the machine assignment and other operation sequence within it.

3.1.2. Selection

The evolution of animal species is formed on the basis of favorable changes and the rejection of undesired changes. Therefore, there is always competition for survival. Offspring who have an advantage over the rest of the species will be survived more. After generating initial population, next stage is to select individuals for reproduction. The selection procedure of each individual is done by the Darwin selection theory, in which each individual survives more permanent with good fitness for generating new offspring. If we select each individual only based on higher fitness, it will be a risk that converges the GA to a local optimal or sub-optimal solution. Conversely, if we select each individual with low fitness, the GA will stay a longer time in promising and feasible region. To overcome this obstacle, the selected individuals (i.e., parents) at random or using the roulette wheel mechanism is chose between populations. In the roulette wheel, more fit chromosomes are more likely for survival.

3.1.3. Fitness function

The fitness of a solution is calculated by synthesizing the two objectives into a weighted sum. The objective values of two criteria have to be normalized before they are summed

because they are of different scales[23] and [27]. Therefore, the fitness of the *l*-thsolution is calculated by:

$$F_T(l) = \left[\alpha_1 \times \frac{f(l)_1 - f_{1_{\min}} + \xi}{f_{1_{\max}} - f_{1_{\min}} + \xi} \right] + \left[\alpha_2 \times \frac{f_2(l) - f_{2_{\min}} + \xi}{f_{2_{\max}} - f_{2_{\min}} + \xi} \right] \quad (2)$$

where $\alpha_1, \alpha_2 \succ 0$ and $\alpha_1 + \alpha_2 = 1$. Also, ξ is negligible amount of greater than zero $(\xi \succ 0)$.For a conventional explanation of the fitness function (2) regards the following parameters:

l : index of chromosomes population($l = 1,2,..., p.$)

$f(l)_i$: value of the *i*-th objective function of the *l*-th chromosome ($i = 1,2,..., q.$)

$f_{i_{\max}}$: maximum value of the *i*-th objective in the current generation.

$f_{i_{\min}}$: minimum value of the *i*-th objective in the current generation.

3.1.4. Crossover operator

The next stage is to combine the selected individuals in order to generate the offspring (i.e., new child), which is called “operator crossover”. This crossover is done on the parents to produce new offspring. Various crossover operators have been suggested in the combinatorial optimization literature, but just several of them are suitable for permutation scheduling. In our study, we use a uniform crossover operator in the GA, which can be described as follows. Two chromosomes are selected from the current population as parents according to the selection process in Section 3.1.2. Suppose that P_1 and P_2 are selected as the parents of the current population, A random mask is created with a length of same as shown Fig.2 for a uniform crossover operator.

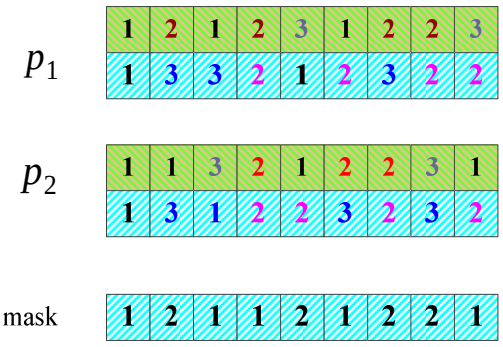

Fig. 2. A random mask for the FJSSP with 3 × 3/9

We define the crossover operator on the operation sequence vector (v_1)whereas genes of v_2 are transmitted together with the operation sequence vector (v_1). The procedures of the

formation of offspring appears in three steps (a, b and c) considering to the produced mask as shown in Fig. 3.

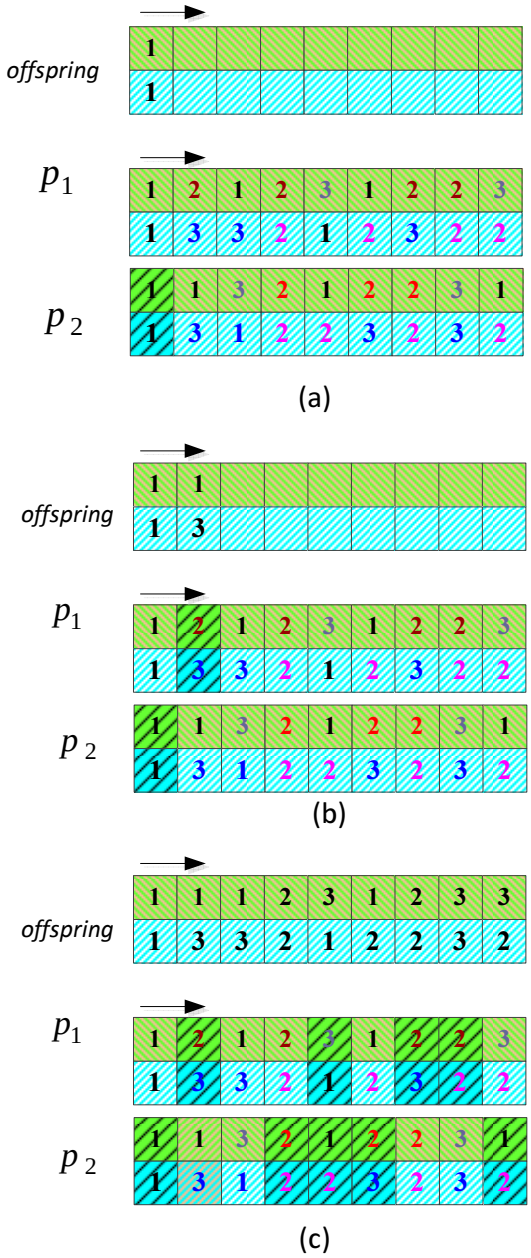

Fig. 3. Offspring production for the FJSSP with 3 × 3/9

3.1.5. Mutation operator

The base of the mutation is to escape from local optimum trap by introducing the faint conversion in sequencing. In this approach, the mutation helps the GA to explore throughout of feasible space and preserves the diversity of population. In our problem, the swap mutation includes two random points in the operation sequence vector (v_1) and performs the whole swap movements existing between two points. First, the swap is performed on the sequence

of vector (v_1) and then the content of v_2 is taken from the parent in the same order, as shown in Fig.4 from the left to right.

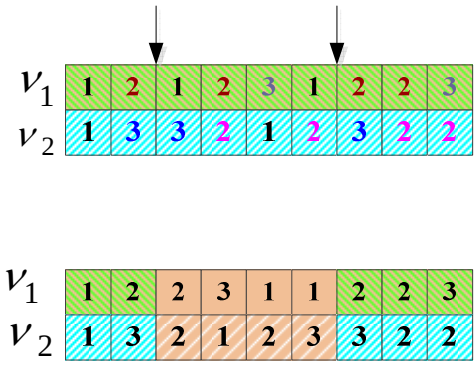

Fig. 4. Mutation of child for the FJSSP with 3× 3/9

An appropriate algorithm should be able to search a larger space in the early generations; in other hands, the algorithm can be found most of the disorganization in the early generations. Then, with the passage of time or in the last generation to be able to converge, therefore, we consider the mutation on the base of an exponential function of generation numbers (i.e., repetitions) in the proposed DGA (see Fig. 5 and Equation 3).

$$f(Gn) = p_{mut} \times e^{-p_{mut}(Gn-1)} \tag{3}$$

p_{mut} : mutation probability

Gn : index of generation number $(Gn = 1,2,...,n)$

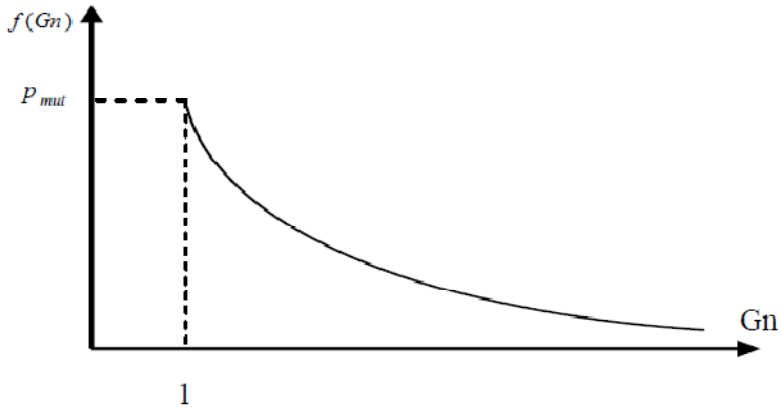

Fig. 5. Mutation rate in each generation

Remark 2: The crossover operator is used for movement to different areas of the feasible space whereas the mutation operator is used to escape from the trap of local optimality.

Remark 3: Note that it will be more desirable when the dispersion rate in a feasible region be larger at the inception of searching process by meta-heuristic algorithms, because it can move

to more areas of feasible region that is less accessible and it searches more spaces. The operation is performed with increasing the crossover operator rate in the GA. However, the purpose of the mutation operator is to search the solutions of available neighbor in the region (i.e., the region found by the crossover operator), in which the mutation operator follows the neighborhood structure of the current solution. Therefore, at the process final generation of meta-heuristic algorithm search, the region of neighborhood should be less. Then, we can conclude that both dispersion rates resulted from the crossover operator and the dispersion rate resulted from the mutation one should be less at the final generation of the meta-heuristic algorithm process. To this end, their possibility quantities should be decreased.

Stopping condition: We consider the stopping condition for the proposed DGA as the generation numbers (i.e., repetition).

3.2. Local search and DGA

The best solution of each generation of the GA (having the lowest fitness function) considered as semi-active solution for the proposed DGA. The semi-active solution should be activated before the application of the DGA. Activation of a solution is realized by local search. We use TSin a certain neighborhood structure as a local search tool. In our problem, local search is applied for machine assignment vector (ν_2) that provides a neighborhood structure. Machines are randomly assigned to the operations. Fig. 6 shows activation of a semi-active solution. Therefore, the solution obtained from TS is considered as an active solution. All of the activated solutions are making up a secondary population of the DGA. The convergence of the second population is performed again by an application of the GA. In addition, Fig. 7 indicates an overview of the proposed DGA.

Fig. 6. Neighborhood structure for a solution

```
-random production of initial population (p) as chromosome
-Calculate the fitness of each chromosome
-Determine the stopping criteria (Gn)
-Do  i<=Gn  (offspring production for new generation)
        -Selection of parents
        -crossover operator
        -mutation operator
 loop
-------------------------------------------------------------------------------------------------
local search:
-identification of semi-active solution in the each of generation
-Application Tabu search as local search
-determine of activation length (l)
 do i<=l
        -activating of semi-active solution
 loop
-Save the activated solution
-creation of secondary population
-goto line 2
```

Fig. 7. Pseudo code of the proposed DGA

4. Numeric examples and comparison

The proposed DGA is implemented in the Visual Basic running on a Pentium IV of 3.0 GHz CPU with 2 GB of RAM. The computational experiments are executed to compare the approaches and to evaluate the efficiency of our proposed DGA. In this section, we present some instances for the evaluation and sensitivity analysis of the algorithm. In addition, we compare the solutions resulted from the DGA with the solutions obtained from other methods (e.g., [12, 13, 16 and 17]), which are among the most recent results in the area of FJSSPs. Experimental simulations are run for several times for each of these test problems.

4.1. Problem 8× 8/27

This is an instance of partial flexibility, the optimization process can be processed a lot of operations on part of the machines. Table1 displays the instance of the medium-scale problem 8× 8/27[19]. We can confirm the validity of our proposed approach by comparing other methods. Let the $f_1(l)$, $f_2(l)$ be the makespan, the total idleness of the l-th solution. The obtained solution by the proposed DGA is shown in the Table 3 with comparing other methods, such as PSO+SA [16], PSO+TS [19], GA+ bottleneck shifting [23]. Fig. 8 shows the optimization solution in the form of a Gantt chart.

$makespen = 14;$

$total \quad idleness = 35;$

4.2. Problem 10×10/30

To evaluate the performance of our proposed algorithm, we apply it for the samples mentioned in Table 2 taken from Kacem et al. [13]. There are 10 jobs with 30 operations to be performed on 10 machines. Fig. 9 shows the optimization solution in the form of a Gantt chart. In addition, the obtained solution by the proposed DGA is shown in Table 3 with comparing other methods.

$makespen = 7;$

$total \quad idleness = 26;$

Table 1. Problem FJSP 8× 8/27.

jobs	$O_{i,j}$	**M 1**	**M 2**	**M 3**	**M 4**	**M 5**	**M 6**	**M 7**	**M 8**
1	$O_{1,1}$	5	3	5	3	3	-	10	9
	$O_{1,2}$	10	-	5	8	3	9	9	6
	$O_{1,3}$	-	10	-	5	6	2	4	5
2	$O_{2,1}$	5	7	3	9	8	-	9	-
	$O_{2,2}$	-	8	5	2	6	7	10	9
	$O_{2,3}$	-	10	-	5	6	4	1	7
	$O_{2,4}$	10	8	9	6	4	7	-	-
3	$O_{3,1}$	10	-	-	7	6	5	2	4
	$O_{3,2}$	-	10	6	4	8	9	10	-
	$O_{3,3}$	1	4	5	6	-	10	-	7
4	$O_{4,1}$	3	1	6	5	9	7	8	4
	$O_{4,2}$	12	11	7	8	10	5	6	9
	$O_{4,3}$	4	6	2	10	3	9	5	7
5	$O_{5,1}$	3	6	7	8	9	-	10	-
	$O_{5,2}$	10	-	7	4	9	8	6	-
	$O_{5,3}$	-	9	8	7	4	2	7	-
	$O_{5,4}$	11	9	-	6	7	5	3	6
6	$O_{6,1}$	6	7	1	4	6	9	-	10
	$O_{6,2}$	11	-	9	9	9	7	6	4
	$O_{6,3}$	10	5	9	10	11	-	10	-
7	$O_{7,1}$	5	4	2	6	7	-	10	-
	$O_{7,2}$	-	9	-	9	11	9	10	5
	$O_{7,3}$	-	8	9	3	8	6	-	10
8	$O_{8,1}$	2	8	5	9	-	4	-	10
	$O_{8,2}$	7	4	7	8	9	-	10	-
	$O_{8,3}$	9	9	-	8	5	6	7	1
	$O_{8,4}$	9	-	3	7	1	5	8	-

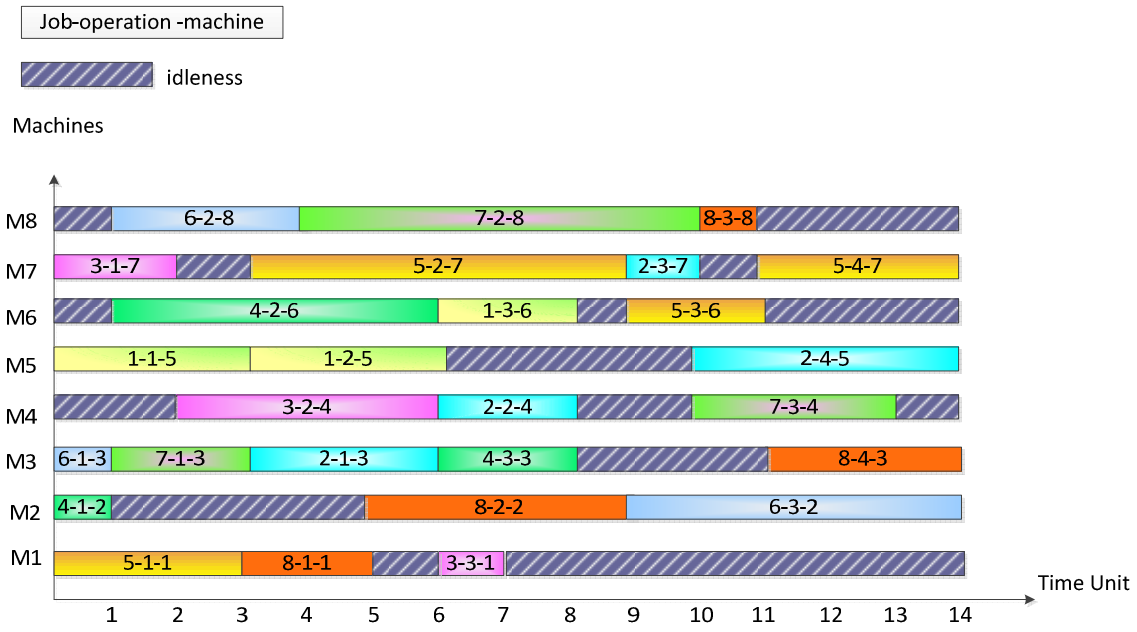

Fig. 8. Optimization solution of problem 8 × 8/27: f_1=14, f_2=35

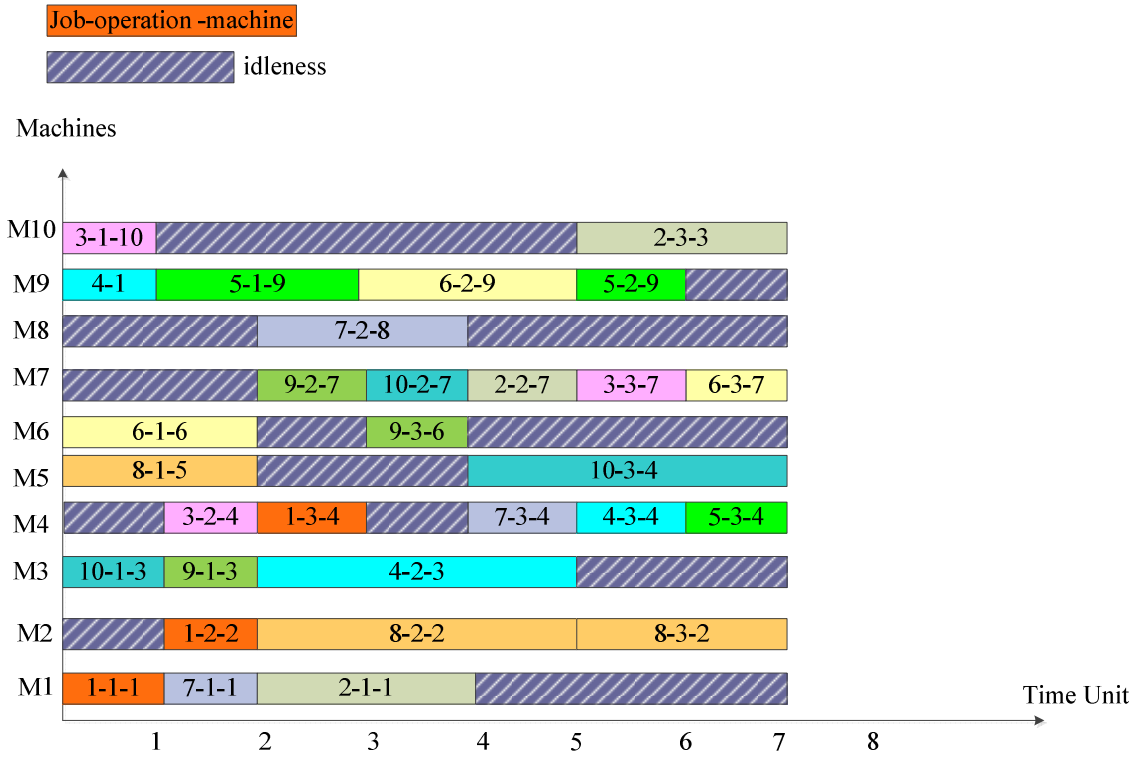

Fig. 9. Optimization solution of problem 10 × 10/30:f_1=8, f_2=26

Table 2. FJSSP 10× 10/30.

job	$O_{i,j}$	**M 1**	**M 2**	**M 3**	**M 4**	**M 5**	**M 6**	**M 7**	**M 8**	**M 9**	**M 10**
1	$O_{1,1}$	1	4	6	9	3	5	2	8	9	5
	$O_{1,2}$	4	1	1	3	4	8	10	4	11	4
	$O_{1,3}$	3	2	5	1	5	6	9	5	10	3
2	$O_{2.1}$	2	10	4	5	9	8	4	15	8	4
	$O_{2,2}$	4	8	7	1	9	6	1	10	7	1
	$O_{2,3}$	6	11	2	7	5	3	5	14	9	2
3	$O_{3,1}$	8	5	8	9	4	3	5	3	8	1
	$O_{3,2}$	9	3	6	1	2	6	4	1	7	2
	$O_{3,3}$	7	1	8	5	4	9	1	2	3	4
4	$O_{4,1}$	5	10	6	4	9	5	1	7	1	6
	$O_{4,2}$	4	2	3	8	7	4	6	9	8	4
	$O_{4,3}$	7	3	12	1	6	5	8	3	5	2
5	$O_{5,1}$	7	10	4	5	6	3	5	15	2	6
	$O_{5,2}$	5	6	3	9	8	2	8	6	1	7
	$O_{5,3}$	6	1	4	1	10	4	3	11	13	9
6	$O_{6,1}$	8	9	10	8	4	2	7	8	3	10
	$O_{6,2}$	7	3	12	5	4	3	6	9	2	15
	$O_{6,3}$	4	7	3	6	3	4	1	5	1	11
7	$O_{7,1}$	1	7	8	3	4	9	4	13	10	7
	$O_{7,2}$	3	8	1	2	3	6	11	2	13	3
	$O_{7,3}$	5	4	2	1	2	1	8	14	5	7
8	$O_{8,1}$	5	7	11	3	2	9	8	5	12	8
	$O_{8,2}$	8	3	10	7	5	13	4	6	8	4
	$O_{8,3}$	6	2	13	5	4	3	5	7	9	5
9	$O_{9,1}$	3	9	1	3	8	1	6	7	5	4
	$O_{9,2}$	4	6	2	5	7	3	1	9	6	7
	$O_{9,3}$	8	5	4	8	6	1	2	3	10	12
10	$O_{10,1}$	4	3	1	6	7	1	2	6	20	6
	$O_{10,2}$	3	1	8	1	9	4	1	4	17	15
	$O_{10,3}$	9	2	4	2	3	5	2	4	10	23

Table 3. Comparison of the proposed DGA with other methods.

Problem ($n\times m$)	Method / Object	PSO+SA		GA+VND	PSO+TS		DGA
		1	2	1	1	2	2
8×8/27	Makespan	15	16	14	15	14	14
	Total work load	75	73	77	75	77	77
	Max. work load	12	13	12	12	12	12
	Total idleness	-	-	-	-	-	35
10×10/30	Makespan	7	-	7	7	-	7
	Total work load	44	-	43	43	-	44
	Max. work load	6	-	5	6	-	6
	Total idleness	-	-	-	-	-	26

5. Conclusions

A modified genetic algorithm, namely duplicate genetic algorithm (DGA), has been offered to solve the bi-objective flexible job shop scheduling problem (FJSSP) that minimizes the makespan and total idleness time. The results have shown that the proposed DGA has offered appropriate solutions compared with other algorithms. In this paper, we have obtained the optimal solution by employing the bi-objective FJSP instead of three-objective FJSSP in the other algorithms. Therefore, to determine the appropriate object, wehave improved the problem solving process and decreased during the problem solving process. Instead of using from total workload machines and critical work load, the total idleness time has used in this paper. Finally, by comparing the computational results, our proposed algorithm has outperformed to other algorithms.

References

[1] Jain, A.S. and Meeran, S. (1998), Deterministic job-shop scheduling: Past, present and future, *European Journal of Operational Research*, Vol. 113, No. 2, pp. 390–434.

[2] Garey, M.R., Johnson, D.S. and Sethi, R. (1996), The complexity of flow shop and job-shop scheduling, *Mathematics of Operations Research*, Vol. 1, No. 2, pp. 117–129.

[3] Bruker, P. and Schlie, R. (1990), Job-shop scheduling with multi-purpose machines, *Computing*, Vol. 45, pp. 369–375.

[4] Brandimarte, P. (1993), Routing and scheduling in a flexible job shop by taboo search, *Annals of Operations Research*, Vol. 41, pp. 157–183.

[5] Hurink, E., Jurisch, B . and Thole, M. (1994), *Tabu* search for the job shop scheduling problem with multi-purpose machine, *Operations Research Spektrum*, Vol. 15, pp. 205–215.

[6] Chambers, J.B. (1996), *Classical and flexible job shop scheduling by tabu search*, Ph.D. thesis, University of Texas at Austin, Austin, USA.

[7] Dauzere-Peres, S. and Paulli, J. (1997), An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search, *Annals of Operations Research*, Vol. 70, pp. 281–306.

[8] Tung, L.F., Lin, L. and Nagi, R. (1999), Multi-objective scheduling for the hierarchical control of flexible manufacturing systems, *The International Journal of Flexible Manufacturing Systems*, Vol. 11, pp. 379–409.

[9] Mastrolilli, M. and Gambardella, L.M. (2000), Effective neighborhood functions for the flexible job shop problem, *Journal of Scheduling*, Vol. 3, No. 1, pp. 3–20.

[10] Mati, Y., Rezg, N. and Xie, X. (2001), An integrated greedy heuristic for a flexible job shop scheduling problem, IEEE International Conference on Systems, Man and Cybernetics, pp. 2534–2539.

[11] Parsopoulos, K.E. and Vrahatis, M.N. (2002), Recent approaches to global optimization problems through particle swarm optimization, *Natural Computing*, Vol. 1, No. (2-3), pp. 235-306.

[12] Kacem, I., Hammadi, S. and Borne, P. (2002a), Approach by localization and multi objective evolutionary optimization for flexible job-shop scheduling problems, IEEE Transactions on Systems, Man and Cybernetics, Part C, Vol. 32, No. 1, pp. 1–13.

[13] Kacem, I., Hammadi, S. and Borne, P. (2002b), Pareto-optimality approach for flexible job-shop scheduling problems Hybridization of evolutionary algrithms and fuzzy logic, *Mathematics and Computers in Simulation*, Vol. 60, pp. 245–276.

[14] Rigao, C. (2004), Tardiness minimization in a flexible job shop: A tabu search approach, *Journal of Intelligent Manufacturing*, Vol. 15, No. 1, pp. 103–115.

[15] Wu, Z. and Weng, M.X. (2005), Multiagent scheduling method with earliness and tardiness objectives in flexible job shops, IEEETransactions on System, Man, and Cybernetics-Part B, Vol. 35, No. 2, pp. 293–301.

[16] Xia, W. and Wu, Z. (2005), An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problem, *Computer & Industrial Engineering*, Vol. 48, pp. 409–425.

[17] Zhang, H. and Gen, M. (2005), Multistage-based genetic algorithm for flexible job-shop scheduling problem, *Journal of Complexity International*, Vol. 11, pp. 223–232.

[18] Gao, J., Sun, L. and Gen, M. (2008), A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, *Computers and Operations Research*, Vol. 35, pp. 2892–2907.

[19] Zhang, G., Shao, X., Li, P. and Gao, L. (2009), An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Computers and Industrial Engineering*, Vol. 56, No. 4, pp. 1309–1318.

[20] Tay, J.C. and Ho, N.B. (2008), Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems, *Computers and Industria Engineering*, Vol. 5, pp. 453–473.

[21] Xing, L.N., Chen, Y.W. and Yang, K.W. (2009a), Multi-objective flexible job shop schedule: design and evaluation by simulation modeling, *Applied Soft Computing*, Vol. 9, pp. 362–376.

[22] Xing, L.N., Chen, Y.W. and Yang, K.W. (2009b), *An* efficient search method for multi objective flexible job shop scheduling problems, *Journal of Intelligent Manufacturing*, Vol. 20, No. 3, pp. 283–293.

[23] Gao, J., Gen, M., Sun, L. and Zhao, X. (2007), A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems, *Computer & industrial Engeenering*, Vol. 53, pp.149–162.

[24] Deb, K. (2001), *Multi-objective optimization using evolutionary algorithms*, Chichester, UK: Wiley.
[25] Hsu, T., Dupas, R., Jolly, D. andGoncalves, G. (2002),Evalution of mutation heuristics for the solving of multiobjective flexible job shop by an evolutionary algorithm, In: Proceedings of the 2002 IEEE international conference on systems, man andcybernetics, pp. 6–9.
[26] Holland, J.H. (1975), *Adaptation in natural and artificial Systems*. Ann Arbor, MI: University of Michigan Press.
[27] Shahsavari pour, N., Modarres, M., Tavakkoli-Moghaddam, R. and Najafi, E. (2010), Optimizing a multi-objectives Time-Cost-Quality trade-off problem by a new hybrid genetic algorithm, *World Applied Sciences Journal*, Vol. 10, No. 3, pp. 355-363.